

***Automatic Methods
for Analyzing Non-Repudiation Protocols
with an Active Intruder***

Francis Klay — Judson Santiago — Laurent Vigneron

N° 6324

Octobre 2007

Thème SYM

 ***apport
de recherche***

Automatic Methods for Analyzing Non-Repudiation Protocols with an Active Intruder

Francis Klay*, Judson Santiago†, Laurent Vigneron‡

Thème SYM — Systèmes symboliques
Projets Cassis

Rapport de recherche n° 6324 — Octobre 2007 — 19 pages

Abstract: Non-repudiation protocols have an important role in many areas where secured transactions with proofs of participation are necessary. Formal methods are clever and without error, therefore using them for verifying such protocols is crucial. In this purpose, we show how to partially represent non-repudiation as a combination of authentications on the Fair Zhou-Gollmann protocol. After discussing its limits, we define a new method based on the handling of the knowledge of protocol participants. This method is very general and is of natural use, as it consists in adding simple annotations, like for authentication problems. The method is very easy to implement in tools able to handle participants knowledge. We have implemented it in the AVISPA Tool and analyzed the optimistic Cederquist-Corin-Dashti protocol, discovering two unknown attacks. This extension of the AVISPA Tool for handling non-repudiation opens a highway to the specification of many other properties, without any more change in the tool itself.

Key-words: cryptographic protocols, non-repudiation, fairness, authentication, automatic analysis, AVISPA Tool

This work is supported by the ANR AVOTÉ.

* France Telecom R&D, francis.klay@orange-ftgroup.com

† DIMap - UFRN, judson@dimap.ufrn.br

‡ LORIA - Nancy Université, laurent.vigneron@loria.fr

Méthodes automatiques pour l'analyse de protocoles de non répudiation en présence d'un intrus actif

Résumé : Les protocoles de non répudiation ont un rôle important dans de nombreux domaines, où des transactions sécurisées avec preuves de participation sont nécessaires. Les méthodes formelles sont astucieuses et sans erreur, il est donc crucial de les utiliser pour vérifier de tels protocoles. Dans ce but, nous montrons sur le protocole Fair Zhou-Gollmann comment représenter partiellement la non répudiation comme la combinaison d'authentifications. Après discussion des limites d'une telle approche, nous définissons une nouvelle méthode basée sur la gestion des connaissances des participants du protocole. Cette méthode est très générale et d'usage naturel, car elle consiste à ajouter de simples annotations, comme pour les problèmes d'authentification. La méthode est très facile à implanter dans des outils capables de gérer les connaissances des participants. Nous l'avons implantée dans l'outil AVISPA et avons analysé le protocole optimiste Cederquist-Corin-Dashti, découvrant deux attaques inconnues jusqu'alors. Cette extension de l'outil AVISPA pour traiter la non répudiation ouvre un boulevard pour spécifier de nombreuses autres propriétés, sans changement supplémentaire dans l'outil.

Mots-clés : protocoles cryptographiques, non répudiation, équité, authentification, analyse automatique, outil AVISPA

1 Introduction

Considering security protocols, the study of properties such as authentication and secrecy has been intensive for years [16], but the interest of other properties such as non-repudiation and fairness has been raised only in the 1990s with the explosion of Internet services and electronic transactions.¹

Non-repudiation protocols are designed for verifying that, when two parties exchange information over a network, neither one nor the other can deny having participated to this communication. Such a protocol must therefore generate evidences of participation to be used in case of a dispute. The basic tools for non-repudiation services have been digital signatures and public key cryptography. Indeed, when one receives a signed message, he has an evidence of the participation and the identity of his party [8].

The majority of the non-repudiation property analysis efforts in the literature are manually driven though. One of the first efforts to apply formal methods to the verification of non-repudiation protocols have been presented by Zhou et al. in [24], where they used SVO logic. In [18] Schneider used process algebra CSP to prove the correctness of a non-repudiation protocol, the well-known Fair Zhou-Gollmann protocol. With the same goal, Bella et al. have used the theorem prover Isabelle [3]. Schneider used a rank function for encoding that in an execution trace, an event happens before another event. The verification is done by analyzing traces in the stable failures models of CSP. Among the automatic analysis attempts, we can cite Shmatikov and Mitchell [19] who have used Mur ϕ , a finite state model-checker, to analyze a fair exchange and two contract signing protocols, Kremer and Raskin [9] who have used a game based model, Armando et al. [2] who used LTL for encoding resilient channels in particular, the very nice work of Gurgens and Rudolph [5] who have used the asynchronous product automata (APA) and the simple homomorphism verification tool (SHVT) [13], raising flaws in three variants of the Fair Zhou-Gollmann protocol and in two fair non-repudiation protocols [7, 22]. Wei and Heather [20] have used FDR, with an approach similar to Schneider, for a variant of the Fair Zhou-Gollmann protocol with timestamps.

The common point between all those works is that they use rich logics, with a classical bad consequence for model checkers, the difficulty to consider large protocols. For avoiding this problem, Wei and Heather [21] used PVS [15], but some of the proof are still manual.

Fairness is more difficult to achieve: no party should be able to reach a point where he has the evidence or the message he requires without the other party also having his required evidence. Fairness is not always required for non-repudiation protocols, but it is usually desirable.

A variety of protocols has been proposed in the literature to solve the problem of fair message exchange with non-repudiation. The first solutions were based on a gradual exchange of the expected information [8]. However this simultaneous secret exchange is troublesome for

¹See <http://www.lsv.ens-cachan.fr/~kremer/FXbib/references.php> for a detailed list of publications related to the analysis of non-repudiation protocols.

actual implementations because fairness is based on the assumption of equal computational power on both parties, which is very unlikely in a real world scenario. A possible solution to this problem is the use of a trusted third party (TTP), and in fact it has been shown that it is impossible to achieve fair exchange without a TTP [14, 12]. The TTP can be used as a delivery agent to provide simultaneous share of evidences. The Fair Zhou-Gollmann protocol [23] is a well known example using a TTP as a delivery agent; a significant amount of work has been done over this protocol and its derivations [3, 6, 18, 24]. However, instead of passing the complete message through the TTP and thus creating a possible bottleneck, recent evolution of protocols resulted in efficient, *optimistic* versions, in which the TTP is only involved in case anything goes wrong. Resolve and abort sub-protocols must guarantee that every party can complete the protocol in a fair manner and without waiting for actions of the other party.

One of these recent protocols is the optimistic Cederquist-Corin-Dashti (CCD) non-repudiation protocol [4]. The CCD protocol has the advantage of not using session labels, contrariwise to many others in the literature [8, 11, 23, 18]. A session label typically consists of a hash of all message components. Gürgens et al. [6] have shown a number of vulnerabilities associated to the use of session labels and, to our knowledge, the CCD protocol is the only optimistic non-repudiation protocol that avoids altogether the use of session labels.

This paper presents a method for automatically verifying non-repudiation protocols in presence of an active intruder. Our method has been implemented in the AVISPA Tool [1]² and we illustrate it with examples. This tool, intensively used for defining Internet security protocols and automatically analyzing their authentication and secrecy properties, did not provide any help for considering non-repudiation properties.

We first consider non-repudiation analysis as a combination of authentication problems, applied to the Fair Zhou-Gollmann protocol. We show the limits of this representation and the difficulties for proving non-repudiation properties using only authentications. Then, we define method based on the analysis of agents knowledge, permitting to handle non-repudiation and fairness properties in a same framework. Our approach is very natural for the user and writing the logical properties is still simple: they correspond to state invariants that are convincing properties for the user. This method is easy to integrate in lazy verification systems, such as the AVISPA Tool, and can also be integrated in any system able to handle agents (or intruder) knowledge. This should permit, contrarily to more complex logics like LTL, to set up abstractions more easily for considering unbounded cases. This should also permit to get a more efficient verification for bounded cases. We illustrate this with the optimistic Cederquist-Corin-Dashti protocol.

2 Non-Repudiation Properties

Non-repudiation (NR) is a general property that may not be clearly defined. It is usually described as a set of required services, depending on the protocol and the required level of

²<http://www.avispa-project.org>

security. In particular, non-repudiation properties may be different whether a trusted third party (TTP) is used or not in the protocol.

Considering a message sent by an originator agent to a recipient agent (possibly via a delivery agent, a TTP), we define below some of the most important non-repudiation services required by most of the existing security applications (for e-commerce for example).

Definition 1 *The service of **non-repudiation of origin**, denoted $\mathcal{NRO}_B(A)$, provides the recipient B with a set of evidences which ensures that the originator A has sent the message. The evidence of origin is generated by the originator and held by the recipient. This property protects the recipient against a dishonest originator.*

Definition 2 *The service of **non-repudiation of receipt**, denoted $\mathcal{NRR}_A(B)$, provides the originator A a set of evidences which ensures that the recipient B has received the message. The evidence of receipt is generated by the recipient and held by the originator. This property protects the originator against a dishonest recipient.*

Definition 3 *The service of **non-repudiation of submission**, denoted $\mathcal{NRS}_A(B)$, provides the originator A a set of evidences which ensures that he has submitted the message for delivery to B . This service only applies when the protocol uses a TTP. Evidence of submission is generated by the delivery agent, and will be held by the originator. This property protects the originator against a dishonest recipient.*

Definition 4 *The service of **non-repudiation of delivery**, denoted $\mathcal{NRD}_A(B)$, provides the originator A a set of evidences which ensures that the recipient B has received the message. This service only applies when the protocol uses a TTP. Evidence of delivery is generated by the delivery agent, and will be held by the originator. This property protects the originator against a dishonest recipient.*

Definition 5 *A service of **fairness** (also called strong fairness) for a non-repudiation protocol provides evidences that if, at the end of the protocol execution, either the originator has the evidence of receipt of the message and the recipient has the evidence of origin of the corresponding message, or none of them has any valuable information. This property protects the originator and the recipient.*

Definition 6 *A service of **timeliness** for a non-repudiation protocol guarantees that, whatever happens during the protocol run, all participants can reach a state that preserves fairness, in a finite time.*

Note that in general sets of evidences such as \mathcal{NRO} , \mathcal{NRR} , \mathcal{NRS} and \mathcal{NRD} are composed with messages signed by an agent.

For the sequel of this paper, we will consider the following definition of an evidence.

Definition 7 *An **evidence** for an agent A for a non-repudiation property P is a message, a part of a message, or a combination of both, received by A that is necessary for guaranteeing property P .*

Note that in this paper, we consider the evidences given by the protocol designer as valid: without intervention of an intruder, those evidences are sufficient to guarantee the non-repudiation service; and in case of a dispute, a judge analyzing them will always be able to protect honest agents.

3 Non-Repudiation as Authentication

It is well known that non-repudiation is a form of authentication [16]. In this section we demonstrate that properties like $\mathcal{NR}\mathcal{O}$, $\mathcal{NR}\mathcal{R}$,... can be at least partially represented by authentication properties. We illustrate this idea with the Fair Zhou-Gollmann protocol. At the end of this section we show strong limitations of this approach in order to motivate the introduction of a new approach in the next section.

3.1 Running Example: the FairZG Protocol

In this section we describe the Fair Zhou-Gollmann protocol (FairZG) [24], a fair non-repudiation protocol that uses a TTP. We have chosen this protocol as a case study to demonstrate our analysis approach because of the existence of significant related work [3, 6, 18]. The protocol is presented below in Alice&Bob notation, where \mathbf{fNRO} , \mathbf{fNRR} , \mathbf{fSUB} and \mathbf{fCON} are labels used to identify the purpose of messages.

1. $A \rightarrow B$: $\mathbf{fNRO.B.L.C.NRO}$
 2. $B \rightarrow A$: $\mathbf{fNRR.A.L.NRR}$
 3. $A \rightarrow \text{TTP}$: $\mathbf{fSUB.B.L.K.SubK}$
 4. $B \leftrightarrow \text{TTP}$: $\mathbf{fCON.A.B.L.K.ConK}$
 5. $A \leftrightarrow \text{TTP}$: $\mathbf{fCON.A.B.L.K.ConK}$
- and $\mathcal{NR}\mathcal{O}_B(A) = \{\mathbf{NRO}, \mathbf{ConK}\}$
 $\mathcal{NR}\mathcal{R}_A(B) = \{\mathbf{NRR}, \mathbf{ConK}\}$

where A (for Alice) is the originator of the message M, B (for Bob) is the recipient of the message M, TTP is the trusted third party, M is the message to be sent from Alice to Bob, C is a commitment (the message M encrypted by a key K), L is a unique session identifier (also called label), K is a symmetric key defined by Alice, NRO is a message used for non-repudiation of origin (the message $\mathbf{fNRO.B.L.C}$ signed by Alice), NRR is a message used for non-repudiation of receipt (the message $\mathbf{fNRR.A.L.C}$ signed by Bob), SubK is a proof of submission of K (the message $\mathbf{fSUB.B.L.K}$ signed by A), ConK is a confirmation of K (the message $\mathbf{fCON.A.B.L.K}$ signed by the TTP).

The main idea of the FairZG protocol is to split the delivery of a message into two parts. First a commitment C, containing the message M encrypted by a key K, is exchanged between Alice and Bob (message \mathbf{fNRO}). Once Alice has an evidence of commitment from Bob (message \mathbf{fNRR}), the key K is sent to a trusted third party (message \mathbf{fSUB}). Once the TTP has received the key, both Alice and Bob can retrieve the evidence \mathbf{ConK} and the key K from the TTP (messages \mathbf{fCON}). This last step is represented by a double direction arrow

in the Alice&Bob notation because it is implementation specific and may be composed by several message exchanges between the agents and the TTP. In this scenario we assume the network will not be down forever and both Alice and Bob have access to the TTP's shared repository where it stores the evidences and the key. This means that the agents will eventually be able to retrieve the key and evidences from the TTP even in case of network failures.

3.2 Non-Repudiation of Origin as Authentication

In our example, the FairZG protocol, non-repudiation of origin should provide the guarantee that if Bob owns \mathcal{NRO} then Alice has sent M to Bob. Proposition 1 shows how this can be partially done with a set of authentications.

Definition 8 *$auth(X, Y, D)$ is the non-injective authentication, and means X authenticates Y on data D .*

The semantics of such a predicate is standard and can be found in [10].

Proposition 1 *Given the FairZG protocol, let B be a honest agent. If $auth(B, A, NRO)$, $auth(B, TTP, ConK)$ and $auth(TTP, A, SubK)$ are satisfied then the non-repudiation service of origin $\mathcal{NRO}_B(A)$ is satisfied.*

Proof: For the two evidences of $\mathcal{NRO}_B(A) = \{NRO, ConK\}$, we have:

- $NRO = \text{Sig}_A(\text{fNRO.B.L.}\{M\}_K)$: since $auth(B, A, NRO)$ is satisfied, there is an agreement on $\text{Sig}_A(\text{fNRO.B.L.C})$ between B and A . From the signature properties this means also an agreement on $\{M\}_K$, thus A has sent $\{M\}_K$.
- $ConK = \text{Sig}_{TTP}(\text{fCON.A.B.L.K})$: as above $auth(B, TTP, ConK)$ implies an agreement on K between B and TTP . Furthermore $SubK = \text{Sig}_A(\text{fSUB, B, L, K})$ thus $auth(TTP, A, SubK)$ implies an agreement on K between TTP and A . By transitivity we have an agreement on K between B and A which means that A has sent K .

As A has sent $\{M\}_K$ and K , he has sent M . The non-injective authentication is only required for $auth(B, TTP, ConK)$ because B can ask many times $ConK$. However since all authentications imply an agreement on the unique session identifier L , this excludes an authentication across different sessions. \square

3.3 Non-Repudiation of Receipt as Authentication

In our example, the FairZG protocol, non-repudiation of receipt should provide the guarantee that if Alice owns \mathcal{NRR} then Bob has receipt M from Alice. Proposition 2 shows how this can be done partially with a set of authentications.

Proposition 2 *Given the FairZG protocol, let B be a honest agent. If $auth(A, B, NRR)$, $auth(A, TTP, ConK)$ and $auth(B, TTP, ConK)$ are satisfied then the non-repudiation service of receipt $\mathcal{NRR}_A(B)$ is satisfied.*

Proof: For the two evidences of $\mathcal{NRR}_A(B) = \{NRR, ConK\}$, we have:

- $NRR = \text{Sig}_B(\text{fNRR.A.L.}\{M\}_K)$: a reasoning as for NRO in Proposition 1 ensures that B has received $\{M\}_K$.
- $ConK = \text{Sig}_{TTP}(\text{fCON.A.B.L.K})$: $\text{auth}(A, TTP, ConK)$ implies an agreement on K between A and TTP. Furthermore $\text{auth}(B, TTP, ConK)$ implies an agreement on K between B and TTP. This means that there is an agreement on K between A and B, thus when A holds ConK, B has received or will be able to receive K.

The proof end is similar to the one of Proposition 1. \square

3.4 Limitations and Difficulties

At this point there are some problems that motivate the introduction of a new approach presented in the next section.

1. If, contrarily to the previous Propositions hypothesis, the evidences owner is dishonest, he can possibly forge a fake evidences set. For example for Bob and \mathcal{NRR} we need to prove that Bob could only own \mathcal{NRR} if Alice has actually sent the correct protocol messages. This may be done as for example in [18], [20] or [6] but this is not trivial.
2. Handling non-repudiation as authentications seems very hard or may not be possible in general. In particular this task seems difficult for optimistic non-repudiation protocols that include sub-protocols like *abort* and *resolve* as presented in the next section.
3. In general verifying Fairness is a delicate stage and the above remarks make this more difficult.

In conclusion, proving non-repudiation with the help of authentications seems for us not to be the right way; this is why in the next section we propose a very easy approach for handling non-repudiation.

4 Non-Repudiation based on Agent Knowledge

In this section, we present a new method for considering non-repudiation services and fairness in a same framework: we introduce a logic permitting to describe states invariants. This logic is a very classical one, except that we define two new predicates, **deduce** and **knows** that permit to consider agents knowledge in the description of goals. The **knows** predicate is also used as protocol annotation, with the semantics *agent X knows (or can deduce) term t*.

4.1 Description of Non-Repudiation Properties

The main role of a non-repudiation protocol is to give evidences of non-repudiation to the parties involved in the protocol. To analyze this kind of protocol, one must verify which participants have their non-repudiation evidences at the end of the protocol execution. For example, if the originator has all its evidences for non-repudiation of receipt, then the service of non-repudiation of receipt is guaranteed. If the recipient has all its evidences for non-repudiation of origin, then the service of non-repudiation of origin is guaranteed. If both parties (or none of them) have their evidences, fairness is guaranteed. In other words, to analyze non-repudiation, we need to verify if a set of terms is known by an agent at the end of the protocol execution.

And for considering a large class of non-repudiation protocols, we shall not restrict evidences to a set of terms, but we have to consider them as a combination of terms using standard logical connectors (conjunction, disjunction, negation).

For considering non-repudiation and fairness properties involving honest and dishonest agents, we have defined a new predicate that permits to access the knowledge of protocol participants. This predicate, named **knows**, is used in the specification of protocol transitions and of properties.

Definition 9 ($\mathcal{NR}_X(Y)$) *Let \mathcal{A} be a set of agents playing a finite number of sessions \mathcal{S} of a protocol, \mathcal{T} a set of terms sent in the messages of this protocol and \mathcal{E} the subset of terms in \mathcal{T} that are part of the evidences of non-repudiation in the protocol. For an agent $X \in \mathcal{A}$, $\mathcal{NR}_X(Y)$ is a logical combination of terms $t \in \mathcal{E}$ that constitute the evidence for a service of non-repudiation \mathcal{NR}_- for agent X wrt. agent Y .*

Definition 10 (**knows**) *Let \mathcal{A} be a set of agents playing a finite number of sessions \mathcal{S} of a protocol, \mathcal{T} a the set of terms. The annotation **knows**(X, s, t) is a predicate with $X \in \mathcal{A}$, $s \in \mathcal{S}$ and $t \in \mathcal{T}$, expressing that agent X , playing in session s of the protocol, knows (or can deduce) the term t .*

The semantics of predicate **knows**(X, s, t) is that the term t can be composed by agent X , according to its current knowledge in the session s of the protocol, whether this agent is honest or not. This composability test can be easily done by any tool that is able to manage agents knowledge or intruder knowledge.

By abuse of notation, we may write **knows**(X, s, L), for a logical formula L combining evidences ($\mathcal{NR}_X(Y)$ for example), considering that the predicate **knows** is an homomorphism:

$$\begin{aligned} \text{knows}(X, s, L_1 \wedge L_2) &= \text{knows}(X, s, L_1) \wedge \text{knows}(X, s, L_2) \\ \text{knows}(X, s, L_1 \vee L_2) &= \text{knows}(X, s, L_1) \vee \text{knows}(X, s, L_2) \\ \text{knows}(X, s, \neg L) &= \neg \text{knows}(X, s, L) \end{aligned}$$

Definition 11 (**deduce**) *Let \mathcal{A} be a set of agents playing a finite number of sessions of a protocol and \mathcal{T} a set of terms. We define **deduce**(X, t), with $X \in \mathcal{A}$ and $t \in \mathcal{T}$, as the predicate which means that X can deduce t from its knowledge.*

We will use the same abuse of notation for **deduce** as for **knows**.

In the following, we assume that each **knows** annotation corresponds to a valid **deduce** predicate on the same information, in order to avoid bad annotations.

Definition 12 *The evidence $\mathcal{NR}_X(Y)$ is **well-formed** if it contains information that uniquely identifies the session, and if it contains an injective function of the message M for which \mathcal{NR}_- acts as a protection against a dishonest agent.*

We now give the results obtained by this representation.

Proposition 3 *Given a non-repudiation service of B against A about a message M with the well-formed evidence $\mathcal{NR}_B(A)$ in session s of a protocol. If the following formulae are true at the session end then the non repudiation service is valid.*

$$\begin{aligned} \text{knows}(B, s, \mathcal{NR}_B(A)) &\Rightarrow \text{knows}(A, s, M) \\ \text{deduce}(B, \mathcal{NR}_B(A)) &\Rightarrow \text{knows}(B, s, \mathcal{NR}_B(A)) \end{aligned}$$

Proof: A sketch of proof is as follows: by the second implication if B is able to deduce $\mathcal{NR}_B(A)$ then $\text{knows}(B, s, \mathcal{NR}_B(A))$ is included in its knowledge. Furthermore since $\mathcal{NR}_B(A)$ is well-formed, $\mathcal{NR}_B(A)$ and $\text{knows}(B, s, \mathcal{NR}_B(A))$ are related to the same session.

Now since $\mathcal{NR}_B(A)$ is well-formed it includes all the information in M , thus the first implication implies an agreement on M between B and A . Finally as $\text{knows}(A, s, M)$ is an annotation, this means that A has followed the protocol, thus he has done what he must do with M . \square

Remark: verifying formulas given in the above Proposition is not a problem, because a priori any theorem prover can compute whatever can be deduced by an agent at a given step of the protocol, especially concerning the **deduce** predicate.

Corollary 1 *Given a non-repudiation service of origin for B against A about message M , in session s of a protocol. If $\mathcal{NRO}_B(A)$ is well-formed and the following formulae are true at the session end then the service is valid.*

$$\begin{aligned} \text{knows}(B, s, \mathcal{NRO}_B(A)) &\Rightarrow \text{knows}(A, s, M) \\ \text{deduce}(B, \mathcal{NRO}_B(A)) &\Rightarrow \text{knows}(B, s, \mathcal{NRO}_B(A)) \end{aligned}$$

Corollary 2 *Given a non-repudiation service of receipt for A against B about message M , in session s of a protocol. If $\mathcal{NRR}_A(B)$ is well-formed and the following formulae are true at the session end then the service is valid.*

$$\begin{aligned} \text{knows}(A, s, \mathcal{NRR}_A(B)) &\Rightarrow \text{knows}(B, s, M) \\ \text{deduce}(A, \mathcal{NRR}_A(B)) &\Rightarrow \text{knows}(A, s, \mathcal{NRR}_A(B)) \end{aligned}$$

4.2 Description of Fairness

In the literature, authors often give different definitions of fairness for non-repudiation protocols. In some definitions none of the parties should have more evidences than the others at any given point in time. Others have a more flexible definition in which none of them should have more evidences than the others in the end of the protocol run. In many works it is also not very clear if only successful protocol runs are taken into account, or partial protocol runs are valid as well.

In this paper the later definition of fairness will be used and we take into account complete protocol runs. By complete protocol runs we mean a run where, even though the protocol could not have reached it's last transition for all agents, there is no executable transition left, i.e. all possible protocol steps have been executed, but this does not mean that all agents are in a final state.

We define this standard fairness as a function of non-repudiation of origin and of non-repudiation of receipt. If both properties, \mathcal{NRO} and \mathcal{NRR} , are ensured or both are not satisfied for a given message M , then we have fairness.

Proposition 4 *Given a protocol whose purpose is to send a message from Alice to Bob, we have the following equivalence concerning the standard definition of fairness for a given session s . If the non-repudiation is valid for the \mathcal{NRO} and \mathcal{NRR} services then:*

$$\text{Fairness} \equiv \text{knows}(\text{Bob}, s, \mathcal{NRO}_{\text{Bob}}(\text{Alice})) \text{ iff } \text{knows}(\text{Alice}, s, \mathcal{NRR}_{\text{Alice}}(\text{Bob}))$$

This result can be generalized to fairness wrt. a set of non-repudiation services as follows.

Theorem 4.1 *Given a protocol involving a finite number of agents, given a finite set of valid non-repudiation services \mathcal{NR} , the protocol is fair wrt. \mathcal{NR} iff*

$$\begin{aligned} \forall \mathcal{NRS}_{1X_1}(Y_1), \mathcal{NRS}_{2X_2}(Y_2) \in \mathcal{NR}, \\ \text{knows}(X_1, s, \mathcal{NRS}_{1X_1}(Y_1)) \text{ iff } \text{knows}(X_2, s, \mathcal{NRS}_{2X_2}(Y_2)) \end{aligned}$$

4.3 Running Example: CCD

For illustrating the analysis method described later on, we will use a recent protocol, the optimistic Cederquist-Corin-Dashti (CCD) non-repudiation protocol [4]. The CCD protocol has been created for permitting an agent A to send a message M to an agent B in a fair manner. This means that agent A should get an evidence of receipt of M by B (EOR) if and only if B has really received M and the evidence of origin from A (EOO). EOR permits A to prove that B has received M , while EOO permits B to prove that M has been sent by A . The protocol is divided into three sub-protocols: the main protocol, an *abort* sub-protocol and a *resolve* sub-protocol.

The Main Protocol. It describes the sending of M by A to B and the exchange of evidences in the case where both agents can complete the entire protocol. If a problem happens to one of the agents, in order to finish properly the protocol, the agents execute the *abort* or the *resolve* sub-protocol with a trusted third party (*TTP*).

The main protocol is therefore composed of the following messages exchanges, described in the Alice&Bob notation:

1. $A \rightarrow B : \{M\}_K.EOO_M$ where $EOO_M = \{B.TTP.H(\{M\}_K).\{K.A\}_{K_{ttp}}\}_{inv(Ka)}$
2. $B \rightarrow A : EOR_M$ where $EOR_M = \{EOO_M\}_{inv(Kb)}$
3. $A \rightarrow B : K$
4. $B \rightarrow A : EOR_K$ where $EOR_K = \{A.H(\{M\}_K).K\}_{inv(Kb)}$

where K is a symmetric key freshly generated by A , H is a one-way hash function, Kg is the public key of agent g and $inv(Kg)$ is the private key of agent g (used for signing messages). Note that we assure that all public keys are known by all agents (including dishonest agents).

In the first message, A sends the message M encrypted by K and the evidence of origin for B (message signed by A , so decryptable by B). In this evidence, B can check his identity, learns the name of the TTP, can check that the hash code is the result of hashing the first part of the message, but cannot decrypt the last part of the evidence; this last part may be useful if any of the other sub-protocols is used.

B answers by sending the evidence of receipt for A , A checking that EOR_M is EOO_M signed by B .

In the third message, A sends the key K , permitting B to discover the message M .

Finally, B sends to A another evidence of receipt, permitting A to check that the symmetric key has been received by B .

The Abort Sub-Protocol. The *abort* sub-protocol is executed by agent A in case he does not receive the message EOR_M at step 2 of the main protocol. The purpose of this sub-protocol is to cancel the messages exchange.

1. $A \rightarrow TTP : \{\mathbf{abort}.H(\{M\}_K).B.\{K.A\}_{K_{ttp}}\}_{inv(Ka)}$
2. $TTP \rightarrow A : \begin{cases} E_{TTP} & \text{where } E_{TTP} = \{A.B.K.H(\{M\}_K)\}_{inv(K_{ttp})} \\ & \text{if } \mathbf{resolved}(A.B.K.H(\{M\}_K)) \\ AB_{TTP} & \text{where } AB_{TTP} = \{A.B.H(\{M\}_K).\{K.A\}_{K_{ttp}}\}_{inv(K_{ttp})} \\ & \text{otherwise} \end{cases}$

In this sub-protocol, A sends to the TTP an abort request, containing the **abort** label and some information about the protocol session to be aborted.

According to what happened before, the TTP has two possible answers: if this is the first problem received by the TTP for this protocol session, the TTP sends a confirmation of abortion, and stores in its database that this protocol session has been aborted; but if the TTP has already received a request for resolving this protocol session, he sends to A the information for completing his evidence of receipt by B .

The Resolve Sub-Protocol. The role of this second sub-protocol is to permit agents A and B to finish the protocol in a fair manner, if the main protocol cannot be run until its end by some of the parties. For example, if B does not get K or if A does not get EOR_K , they can invoke the *resolve* sub-protocol.

1. $G \rightarrow TTP : EOR_M$
2. $TTP \rightarrow G : \begin{cases} AB_{TTP} & \text{if } \text{aborted}(A.B.K.H(\{M\}_K)) \\ E_{TTP} & \text{otherwise} \end{cases}$

where G stands for A or B .

A resolve request is done by sending EOR_M to the TTP. If the protocol session has already been aborted, the TTP answers by the abortion confirmation. If this is not the case, the TTP sends E_{TTP} so that the user could complete its evidence of receipt (if G is A) or of origin (if G is B). Then the TTP stores in its database that this protocol session has been resolved.

Agents' Evidences. For this protocol, according to [4], the logical formulas of evidences are:

$$\begin{aligned} \mathcal{NRO}_B(A) &= \{M\}_K \wedge EOO_M \wedge K \\ \mathcal{NRR}_A(B) &= \{M\}_K \wedge EOR_M \wedge (EOR_K \vee E_{TTP}) \end{aligned}$$

Note that there are two possibilities of evidences for non-repudiation of receipt, according to the way the protocol is run.

According to our method, we simply have to annotate protocol steps with **knows** predicates, and then write the logical formula to verify. The following table shows where those annotations take place in the three CCD sub-protocols, for considering non-repudiation of origin and of receipt.

$\mathcal{NRO}_B(A)$	Protocol - step	$\mathcal{NRR}_A(B)$	Protocol - step
knows ($B, s, \{M\}_K$)	Main - 1.	knows ($A, s, \{M\}_K$)	Main - 1.
knows (B, s, EOO_M)	Main - 1.	knows (A, s, EOR_M)	Main - 2.
knows (B, s, K)	Main - 3.	knows (A, s, EOR_K)	Main - 4.
knows (B, s, K)	Resolve - 2.	knows (A, s, E_{TTP})	Abort - 2.
		knows (A, s, E_{TTP})	Resolve - 2.

According to Corollary 1, **non-repudiation of origin** for the CCD protocol is represented by the following invariant formulas:

$$\begin{aligned} \text{knows}(B, s, \{M\}_K \wedge EOO_M \wedge K) &\Rightarrow \text{knows}(A, s, M) \\ \text{deduce}(B, \{M\}_K \wedge EOO_M \wedge K) &\Rightarrow \text{knows}(B, s, \{M\}_K \wedge EOO_M \wedge K) \end{aligned}$$

According to Corollary 2, **non-repudiation of receipt** for the CCD protocol is represented by the following invariant formulas:

$$\begin{aligned} & \text{aknows}(A, s, \{M\}_K \wedge EOR_M \wedge (EOR_K \vee E_{TTP})) \Rightarrow \text{aknows}(B, s, M) \\ & \text{deduce}(A, s, \{M\}_K \wedge EOR_M \wedge (EOR_K \vee E_{TTP})) \\ & \quad \Rightarrow \text{aknows}(A, s, \{M\}_K \wedge EOR_M \wedge (EOR_K \vee E_{TTP})) \end{aligned}$$

For analyzing **fairness**, this protocol requires timeliness, that is each participant should reach a final state before testing fairness. Fairness for the CCD protocol is described by the following logical formulas, a very simple application of Theorem 4.1:

$$\text{aknows}(A, s, \mathcal{NRR}_A(B)) \Leftrightarrow \text{aknows}(B, s, \mathcal{NRO}_B(A))$$

Basically the property states that if A knows the EOR evidence ($\{M\}_K$, EOR_M , and EOR_K or E_{TTP}), then B must know the EOO evidence. And symmetrically for B , if B knows the EOO evidence ($\{M\}_K$, EOO_M , and K or E_{TTP}), then A must know the EOR evidence.

The CCD protocol has been specified in the AVISPA Tool, with the description of the fairness property given above. The detailed formulas used in the AVISPA Tool, with an LTL syntax, are:

$$\begin{aligned} & \square \left(\left(\begin{array}{l} \text{aknows}(A, s, \{M\}_K) \wedge \\ \text{aknows}(A, s, EOR_M) \wedge \\ (\text{aknows}(A, s, EOR_K) \vee \text{aknows}(A, s, E_{TTP})) \end{array} \right) \Rightarrow \left(\begin{array}{l} \text{aknows}(B, s, \{M\}_K) \wedge \\ \text{aknows}(B, s, EOO_M) \wedge \\ \text{aknows}(B, s, K) \end{array} \right) \right) \\ & \square \left(\left(\begin{array}{l} \text{aknows}(B, s, \{M\}_K) \wedge \\ \text{aknows}(B, s, EOO_M) \wedge \\ \text{aknows}(B, s, K) \end{array} \right) \Rightarrow \left(\begin{array}{l} \text{aknows}(A, s, \{M\}_K) \wedge \\ \text{aknows}(A, s, EOR_M) \wedge \\ (\text{aknows}(A, s, EOR_K) \vee \text{aknows}(A, s, E_{TTP})) \end{array} \right) \right) \end{aligned}$$

Several scenarios have been run, and two of them have raised an attack, showing that the CCD protocol does not provide the fairness property for which it has been designed.

The first attack has been found for a scenario where only one session of the protocol is run, between honest agents. The problem is raised when some messages of the main protocol are delayed, either by a slow network traffic or by the action of an intruder. The consequence of this delay is that A will invoke the *abort* sub-protocol and B will invoke the *resolve* sub-protocol. And if the resolve request reaches the TTP before the abort request, B will get all his necessary evidences from the TTP, while A is not able to get all his evidences even with the help of the TTP.

The originality of this attack is that, at the end:

- A will guess (according to the answer received to his abort request) that the protocol has been resolved by B , so he will assume that B knows M and can build the proof that A has sent it; but A cannot prove this;
- B has resolved the protocol and has received from the TTP the information for getting M and building the proof that A has sent M ; but he does not know that A does not have his proof;

- the TTP will think that B has asked for the protocol to be resolved, followed by A ; so for him, both A and B can build their evidences.

So, this trace shows that the CCD protocol is not fair, even if both agents A and B are honest. The attack is due to a malicious intruder or a network problem, and the TTP is of no help for detecting the problem.

The second attack is a variant: it happens when agent A plays the protocol with a dishonest agent B (named i , for *intruder*). As soon as i has received the first message from A , he builds EOR_M and sends it to the TTP as resolve request. When A decides to abort the protocol, this is too late: the protocol has already been resolved, the intruder can get M and build the proof that A has sent M , and A cannot build the evidence of receipt. We have corrected the protocol and the numerous scenarios that have been tried on the new version have not raised any attack. This experiment on the CCD protocol is detailed in [17].

5 Conclusion

Non-repudiation protocols have an important role in many areas where secured transactions with proofs of participation are necessary. The evidences of origin and receipt of a message are two examples of elements that the parties should have at the end of the communication. We have given two very different examples of such protocols. The FairZG protocol is an intensively studied protocol in which the role of the trusted third party is essential. The CCD protocol is a more recent non-repudiation protocol that avoids the use of session labels and distinguishes itself by the use of an optimistic approach, the trusted third party being used only in case of a problem in the execution of the main protocol.

The fairness of a non-repudiation protocol is a property difficult to analyze and there are very few tools that can handle the automatic analysis of this property. The contribution of this work is twofold. First, we have illustrated with the FairZG protocol how difficult it is to consider full non-repudiation properties using only a combination of authentications.

Second, we have defined a new method that permits to handle in a very easy way non-repudiation properties and fairness in a same framework. This method is based on the handling of agents knowledge and can be used to automatically analyze non-repudiation protocols as well as contract signing protocols [19]. We have implemented it in the AVISPA Tool and have successfully applied it to the CCD protocol, proving that it is not fair. We have also tested other specifications of the CCD protocol, for example with secure communication channels between agents and the TTP, and for the original definition for the *abort* sub-protocol: no attack has been found; but using such channels is not considered as acceptable, because it requires too much work for the TTP.

Our method, based on the writing of simple state invariants, is of easy use, and can be implemented in any tool handling agents (or intruder) knowledge. It should be very helpful for setting abstractions for handling unbounded scenarios, and it should be very efficient for bounded verifications, as it has been the case in our implementation. We hope that this work will open a highway to the specification of many other properties, without any more change in the specification languages and the analysis engines.

References

- [1] A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In K. Etessami and S. K. Rajamani, editors, *Computer Aided Verification, 17th International Conference, CAV 2005*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285, Edinburgh, Scotland, UK, 2005. Springer.
- [2] A. Armando, R. Carbone, and L. Compagna. LTL Model Checking for Security Protocols. In *20th IEEE Computer Security Foundations Symposium (CSF'07)*, pages 385–396, Los Alamitos, CA, USA, 2007. IEEE Computer Society.
- [3] G. Bella and L. C. Paulson. Mechanical Proofs about a Non-repudiation Protocol. In R. J. Boulton and P. B. Jackson, editors, *Theorem Proving in Higher Order Logics, 14th International Conference, TPHOLs 2001*, volume 2152 of *Lecture Notes in Computer Science*, pages 91–104, Edinburgh, Scotland, UK, 2001. Springer.
- [4] J. Cederquist, R. Corin, and M. T. Dashti. On the Quest for Impartiality: Design and Analysis of a Fair Non-repudiation Protocol. In S. Qing, W. Mao, J. Lopez, and G. Wang, editors, *Information and Communications Security, 7th International Conference, ICICS 2005*, volume 3783 of *Lecture Notes in Computer Science*, pages 27–39, Beijing, China, 2005. Springer.
- [5] S. Gürgens and C. Rudolph. Security Analysis of Efficient (Un-)fair Non-repudiation Protocols. *Formal Aspects of Computing*, 17(3):260–276, 2005.
- [6] S. Gürgens, C. Rudolph, and H. Vogt. On the Security of Fair Non-repudiation Protocols. In C. Boyd and W. Mao, editors, *Information Security, 6th International Conference, ISC 2003*, volume 2851 of *Lecture Notes in Computer Science*, pages 193–207, Bristol, UK, 2003. Springer.
- [7] S. Kremer and O. Markowitch. Optimistic Non-repudiable Information Exchange. In J. Biemond, editor, *21st Symp. on Information Theory in the Benelux*, pages 139–146, Wassenaar (NL), 2000. Werkgemeenschap Informatieen Communicatietheorie, Enschede (NL).
- [8] S. Kremer, O. Markowitch, and J. Zhou. An Intensive Survey of Fair Non-repudiation Protocols. *Computer Communications*, 25(17):1606–1621, 2002.
- [9] S. Kremer and J.-F. Raskin. A Game-Based Verification of Non-repudiation and Fair Exchange Protocols. In K. G. Larsen and M. Nielsen, editors, *CONCUR 2001 - Concurrency Theory, 12th International Conference*, volume 2154 of *Lecture Notes in Computer Science*, pages 551–565, Aalborg, Denmark, 2001. Springer.

- [10] G. Lowe. A Hierarchy of Authentication Specification. In *10th Computer Security Foundations Workshop (CSFW'97)*, pages 31–44, Rockport, Massachusetts, USA, 1997. IEEE Computer Society.
- [11] O. Markowitch and S. Kremer. An Optimistic Non-repudiation Protocol with Transparent Trusted Third Party. In G. I. Davida and Y. Frankel, editors, *Information Security, 4th International Conference, ISC 2001*, volume 2200 of *Lecture Notes in Computer Science*, pages 363–378, Malaga, Spain, 2001. Springer.
- [12] O. Markowitch and Y. Roggeman. Probabilistic Non-Repudiation without Trusted Third Party. In *Second Workshop on Security in Communication Networks'99*, Amalfi, Italy, 1999.
- [13] P. Ochsenschläger, J. Repp, R. Rieke, and U. Nitsche. The SH-Verification Tool - Abstraction-Based Verification of Co-operating Systems. *Formal Aspects of Computing*, 10(4):381–404, 1998.
- [14] H. Pagnia and F. C. Gärtner. On the Impossibility of Fair Exchange without a Trusted Third Party. Technical Report TUD-BS-1999-02, Darmstadt University of Technology, Darmstadt, Germany, 1999.
- [15] A. W. Roscoe, C. A. R. Hoare, and R. Bird. *The Theory and Practice of Concurrency*. Prentice Hall PTR, 1997.
- [16] P. Ryan, M. Goldsmith, G. Lowe, B. Roscoe, and S. Schneider. *Modelling and Analysis of Security Protocols*. Addison Wesley, 2000.
- [17] J. Santiago and L. Vigneron. Optimistic Non-repudiation Protocol Analysis. In D. S. et al., editor, *Proceedings of the Workshop in Information Security Theory and Practices (WISTP'2007), Smart Cards, Mobile and Ubiquitous Computing Systems*, volume 4462 of *Lecture Notes in Computer Science*, pages 90–101, Heraklion (Greece), May 2007. Springer.
- [18] S. Schneider. Formal Analysis of a Non-Repudiation Protocol. In *Proceedings of The 11th Computer Security Foundations Workshop*, pages 54–65. IEEE Computer Society Press, 1998.
- [19] V. Shmatikov and J. C. Mitchell. Analysis of Abuse-Free Contract Signing. In Y. Frankel, editor, *Financial Cryptography, 4th International Conference, FC 2000*, volume 1962 of *Lecture Notes in Computer Science*, pages 174–191, Anguilla, British West Indies, 2000. Springer.
- [20] K. Wei and J. Heather. Towards Verification of Timed Non-repudiation Protocols. In T. Dimitrakos, F. Martinelli, P. Y. A. Ryan, and S. A. Schneider, editors, *Formal Aspects in Security and Trust, 3rd international Workshop, FAST 2005, Revised selected papers*, volume 3866 of *Lecture Notes in Computer Science*, pages 244–257, Newcastle, UK, July 2006. Springer.

- [21] K. Wei and J. Heather. A Theorem-Proving Approach to Verification of Fair Non-repudiation Protocols. In T. Dimitrakos, F. Martinelli, P. Y. A. Ryan, and S. A. Schneider, editors, *Formal Aspects in Security and Trust, 4th international Workshop, FAST 2006, Revised selected papers*, volume 4691 of *Lecture Notes in Computer Science*, pages 202–219, Hamilton, Ontario, Canada, Aug. 2007. Springer.
- [22] J. Zhou, R. H. Deng, and F. Bao. Evolution of Fair Non-repudiation with TTP. In *ACISP '99: Proceedings of the 4th Australasian Conference on Information Security and Privacy*, volume 1587 of *Lecture Notes in Computer Science*, pages 258–269. Springer-Verlag, 1999.
- [23] J. Zhou and D. Gollmann. A Fair Non-repudiation Protocol. In *1996 IEEE Symposium on Security and Privacy*, pages 55–61, Oakland, CA, USA, 1996. IEEE Computer Society.
- [24] J. Zhou and D. Gollmann. Towards verification of non-repudiation protocols. In *Proceedings of 1998 International Refinement Workshop and Formal Methods Pacific*, pages 370–380, Canberra, Australia, September 1998.

Contents

1	Introduction	3
2	Non-Repudiation Properties	4
3	Non-Repudiation as Authentication	6
3.1	Running Example: the FairZG Protocol	6
3.2	Non-Repudiation of Origin as Authentication	7
3.3	Non-Repudiation of Receipt as Authentication	7
3.4	Limitations and Difficulties	8
4	Non-Repudiation based on Agent Knowledge	8
4.1	Description of Non-Repudiation Properties	9
4.2	Description of Fairness	11
4.3	Running Example: CCD	11
5	Conclusion	15



Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
<http://www.inria.fr>
ISSN 0249-6399